Solidigm™ Data Center SSD

Troubleshooting and Health Monitoring Guide

White Paper

August 2025

Revision 001

SOLIDIGM™

Revision History

Revision	Description	Revision Date
001	Initial release	June 2025

All information provided here is subject to change without notice.
The products described in this document may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
Solidigm technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer.
Solidigm disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase.
Cost reduction scenarios described are intended as examples of how a given Solidigm-based product, in the specified circumstances and configurations, may affect future costs and provide cost -savings. Circumstances will vary. Solidigm does not guarantee any costs or cost reduction.
Solidigm does not control or audit the design or implementation of third-party benchmark data or Web sites referenced in this document.
Solidigm encourages all of its customers to visit the referenced Web sites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate and reflect performance of systems available for purchase.
© Solidigm. Solidigm and the Solidigm logo are trademarks of Solidigm in the United States and other countries. Other names and brands may be claimed as the property of others.

Contents

1. Introduction

This guide outlines best practices for troubleshooting and healthy monitoring of Solidigm™ Data Center SSDs using the Solidigm™ Storage Tool or the open-source NVMe-CLI tool.

1.1 Overview

This document aims to assist customers in utilizing the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool for health monitoring and troubleshooting of Solidigm™ Data Center SSDs. These tools enable customers to create custom health management software for centralized datacenter management. It is designed for IT administrators, engineers, solutions architects, and field sales professionals.

1.2 Applicable Solidigm™ Data Center SSDs

The following SSDs are verified or supported:

- Solidigm™ SSD D7-P5510 series
- Solidigm™ SSD D7-P5520 and D7-P5620 series
- Solidigm™ SSD D7-P5810 series
- Solidigm™ SSD D5-P5316 series
- Solidigm™ SSD D5-P5336 series
- Solidigm™ SSD D7-PS1010 and D7-PS1030 series

1.3 Compatible Operating Systems

The tests in this document were conducted on Rocky Linux release 8.10 with the latest Linux Kernel available at the time of initial publication. Comparable results are expected on the following operating systems with default NVMe drivers:

- Red Hat Enterprise Linux 8.10 or later
- Rocky Linux release 8.10 or later
- SUSE Linux Enterprise Server 15 SP5 or later
- Windows 2019
- Ubuntu 24.04 LTS or later

Note: The Windows in-box NVMe driver may not support some commands as expected.

2. Solidigm™ Storage Tool (SST) and Open-sourced NVMe-CLI Tool

To implement these best practices for troubleshooting and healthy monitoring, you can use either the Solidigm™ Storage Tool or the open-source NVMe-CLI tool.

2.1 Solidigm™ Storage Tool (SST)

The Solidigm™ Storage Tool (SST) facilitates the management of Solidigm™ SSDs. Its Graphical User Interface (GUI) is available exclusively for Windows*. The Command-Line Interface (CLI) supports Windows*, Linux*, and ESXi*.

SST provides access to drive details, health status, SMART attributes, firmware updates, diagnostic scans, and secure erase functions.

Download the appropriate version of the Solidigm™ Storage Tool from the following page:
https://www.solidigm.com/support-page/drivers-downloads/ka-00085.html

2.2 Open-Sourced NVMe-CLI Tool

NVM Express® (NVMe®) technology supports a comprehensive set of industry-standard software, drivers, and management tools for storage. The NVMe Command Line Interface (NVMe-CLI) is designed to manage NVMe SSDs on Linux.

NVMe-CLI is available as a package for all major Linux distributions. Installation instructions for various distributions, such as Ubuntu and Debian, are available on the GitHub page:
https://github.com/linux-nvme/nvme-cli

Note: NVMe-CLI version 2.11 or later includes the Solidigm NVMe-CLI plug-in, which enables access to Solidigm-specific readings and logs.

2.3 Retrieve SSDs information by using tools

To support troubleshooting and healthy monitoring, you can use the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool to retrieve detailed information about Solidigm™ Data Center SSDs installed in your system

Using Solidigm™ Storage Tool (SST)
Run the following command to display details for a specific SSD:

Example:

```
# sst show -ssd 0
```

Sample Output:

```
- BTAX135003JL7P6DGN 1 -

Bootloader : Value not found
Capacity : 7.68 TB (7,680,000,000,000 bytes)
DevicePath : /dev/nvme0n1
DeviceStatus : Healthy
```

```
Firmware : 9CV10510
FirmwareUpdateAvailable : The selected drive contains current firmware as of this tool release.
Index : 0
MaximumLBA : 14999999999
ModelNumber : INTEL SSDPF2KX076T1
NamespaceId : 1
PercentOverProvisioned : 99.98
ProductFamily : Intel SSD DC P5520 Series
SMARTEnabled : True
SectorDataSize : 512
SerialNumber : BTAX135003JL7P6DGN
```

This output includes the SSD's serial number, model number, capacity, firmware version, and health status.

Using NVMe-CLI Tool
The NVMe-CLI tool provides similar SSD details in a different format. Use the following command to list installed SSDs:

Example:

```
# nvme list |grep nvme0
/dev/nvme0n1   /dev/ng0n1    BTAX135003JL7P6DGN   INTEL SSDPF2KX076T1    0x1   7.68  TB /   7.68  TB
   512   B +  0 B   9CV10510
```

This output displays the SSD's serial number, model number, capacity, and firmware version.

Accessing SMART Data and Solidigm-Specific Logs
Both SST and NVMe-CLI (with the Solidigm plug-in) can retrieve extensive SMART data, including the vendor-specific CA log page, in a human-readable format.

SST Example:

```
# sst show -all -ssd 0
…
```

NVMe-CLI with Solidigm Plug-in Example:

```
# nvme solidigm smart-log-add /dev/nvme0n1
…
```

These commands provide detailed SMART information, including Solidigm's CA log page, to support healthy monitoring and troubleshooting.

3. SMART Reading and Health Check

This section outlines how to perform healthy monitoring and assess the health status of Solidigm™ Data Center SSDs using the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool.

3.1 Healthy Indicators

To check the overall health status of an SSD, use the following commands to retrieve key indicators.

Using Solidigm™ Storage Tool (SST)
Run the following command to display the health status of a specific SSD:

Example:

```
# sst show -ssd 0 |grep DeviceStatus
```

Sample Output:

```
DeviceStatus : Healthy
```

Using the NVMe-CLI tool
Run the following command to retrieve the health status of an SSD:

Example:

```
# nvme solidigm id-ctrl /dev/nvme0 |grep health
```

Sample Output:

```
health    : healthy
```

3.2 SMART Log

To retrieve SMART attributes (Log Identifier 02h) for healthy monitoring and troubleshooting, use the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool with the following commands.

Using Solidigm™ Storage Tool (SST):
Run the following command to display SMART and health information for a specific SSD:
Example:

```
# sst show -ssd 0 -nvmelog 2
```

Sample Output:

```
-  BTAX135003JL7P6DGN -

- NVMeLog SMART and Health Information -

Volatile memory backup device has failed : False
Temperature has exceeded a critical threshold : False
Temperature - Celsius : 41
Device reliability has degraded : False
Media is in a read-only mode : False
Available Spare Normalized percentage of the remaining spare capacity available : 100
Host Write Commands : 0x12A7B17F67
Available Spare Threshold Percentage : 10
Controller Busy Time : 0x0B38
Power Cycles : 0x29
```

```
Data Units Read : 0xC0FA32F9
Media Errors : 0x0
Host Read Commands : 0x1CD24EA5C7
Available Spare Space has fallen below the threshold : False
Critical Warnings : 0
Data Units Written : 0x93171E96
Number of Error Info Log Entries : 0x0
Unsafe Shutdowns : 0x13
Power On Hours : 0x16AD
Percentage Used : 3
```

This output provides key health metrics, such as temperature, spare capacity, and error counts, to support healthy monitoring.

Using NVMe-CLI Tool

Run the following command to retrieve SMART log data for an SSD:

Example:

```
# nvme smart-log /dev/nvme0
```

Sample Output:

```
Smart Log for NVME device:nvme0 namespace-id:ffffffff
critical_warning                    : 0
temperature                         : 105 °F (314 K)
available_spare                     : 100%
available_spare_threshold           : 10%
percentage_used                     : 3%
endurance group critical warning summary: 0
Data Units Read                     : 3237622521 (1.66 PB)
Data Units Written                  : 2467765910 (1.26 PB)
host_read_commands                  : 123787453895
host_write_commands                 : 80122838887
controller_busy_time                : 2872
power_cycles                        : 41
power_on_hours                      : 5805
unsafe_shutdowns                    : 19
media_errors                        : 0
num_err_log_entries                 : 0
Warning Temperature Time            : 0
Critical Composite Temperature Time     : 0
Thermal Management T1 Trans Count       : 0
Thermal Management T2 Trans Count       : 0
Thermal Management T1 Total Time        : 0
Thermal Management T2 Total Time        : 0
```

This output includes health metrics like temperature, media errors, and usage statistics, aiding in healthy monitoring and troubleshooting.

3.3 Health Monitoring

This subsection describes key indicators for healthy monitoring of Solidigm™ Data Center SSDs using the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool. These indicators help assess SSD health and identify potential issues.

- **"Critical_Warning"** Count

"Critical Warning" SMART attribute flags conditions that may indicate potential SSD issues, including:

- Available Spare is below Threshold
- Temperature has exceeded Threshold

- Reliability is degraded due to excessive media or internal errors
- Media is placed in Read-Only Mode
- Volatile Memory Backup System has failed (e.g., enhanced power loss capacity test failure)

Note: Even if the drive's health status (per NVMe Identify Controller) shows "Healthy," a Critical Warning flag may still be set. Regularly monitor this counter to detect pre-failure conditions. If the Critical Warning count exceeds the predefined threshold, system administrators should consider actions such as hardware replacement.

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -ssd 0 -sensor |grep DeviceStatus
```

Sample Output:

```
DeviceStatus : Healthy
```

Note: SST uses "DeviceStatus" instead of "Critical Warning" to indicate health status.

Using the NVMe-CLI tool:
Example:

```
# nvme smart-log /dev/nvme0 | grep critical_warning
```

Sample Output:

```
critical_warning                : 0
```

- **"Available Spare"** Indicator

The "Available Spare" indicator shows the percentage of remaining spare capacity. If this value reaches 0%, contact a Solidigm representative for potential hardware replacement.

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -ssd 0 -sensor |grep -w AvailableSpare
```

Sample Output:

```
AvailableSpare : 100
```

Using NVMe-CLI Tool
Example:

```
# nvme smart-log /dev/nvme0 |grep -w available_spare
```

Sample Output:

```
available_spare                 : 100%
```

- **"Percentage Used"** Indicator

The "Percentage Used" indicator reflects the endurance of Solidigm™ Data Center SSDs. It starts at 0% when the SSD is first installed and may remain at 0% initially, gradually increasing to a maximum of 100%. For details on this behavior, refer to the Technical Advisory:
https://www.solidigm.com/support-page/maintenance-tools/ka-01469.html

Using Solidigm™ Storage Tool (SST):

Example:

```
# sst show -ssd 0 -sensor |grep PercentageUsed
```

Sample Output:

```
- PercentageUsed : 2
```

Using the NVMe-CLI tool:
Example:

```
# nvme smart-log /dev/nvme0 | grep percentage_used
```

Sample Output:

```
percentage_used                 : 0%
```

- **"Unsafe Shutdowns"** Indicator

The "Unsafe Shutdowns" indicator tracks unexpected power loss events, which may depend on server system stability, HSBP reliability, or hot-plug events. Monitor this indicator as needed.

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -ssd 0 -sensor |grep UnsafeShutdowns
```

Sample Output:

```
UnsafeShutdowns : 0x05
```

Using the NVMe-CLI tool:
Example:

```
# nvme smart-log /dev/nvme0 | grep unsafe_shutdowns
```

Sample Output:

```
unsafe_shutdowns                : 31
```

- **"Media Errors"** Indicator

The "Media Errors" indicator counts unrecovered data integrity errors detected by the controller, such as uncorrectable ECC, CRC checksum failures, or LBA tag mismatches. Combine this with "DeviceStatus" and "Critical Warning" indicators to assess SSD health. If the count increases significantly (consult Solidigm's global call center for specific thresholds), central management software should take action based on predefined thresholds.

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -ssd 0 -sensor |grep MediaErrors
```

Sample Output:

```
MediaErrors : 0x00000000000000000
```

Using the NVMe-CLI tool:
Example:

```
# nvme smart-log /dev/nvme0 | grep media_errors
```

Sample Output:

```
media_errors                    : 0
```

- **"End-to-End Error Detection Count"** Indicator

The "End-to-End Error Detection Count" reports errors detected and corrected by the SSD's hardware.

Using Solidigm™ Storage Tool (SST):
Example:

```
# show -ssd 0 -sensor |grep EndToEnd
```

Sample Output:

```
EndToEndErrorDetectionCount : 0
```

Using the NVMe-CLI tool:
Example:

```
# nvme solidigm smart-log-add /dev/nvme1 |grep e2e
```

Sample Output:

```
0xb8    e2e_error_detect_count                  100       0
```

- **Temperature Reading and Statistics**

The "Temperature" indicator displays the SSD's composite temperature, and the Temperature Statistics log is a Solidigm-specific log page. Use the following commands to access these readings:

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -ssd 0 -sensor |grep Temperature
```

Sample Output:

```
HighestLifetimeTemperature : 69
LowestLifetimeTemperature : 21
TemperatureThresholdExceeded : False
Temperature - Celsius : 41
```

Using the NVMe-CLI tool:
Example:

```
# nvme solidigm temp-stats /dev/nvme0
```

Sample Output:

```
Current temperature        : 40
Last critical overtemp flag : 0
Life critical overtemp flag : 0
Highest temperature        : 69
Lowest temperature         : 21
Max operating temperature   : 70
Min operating temperature   : 0
Estimated offset           : 0
```

Note: The maximum operating temperature corresponds to the "Thermal Throttle Start" temperature in the Solidigm™ Data Center SSD product specification. Access to the product specification requires a signed NDA with Solidigm. Contact a Solidigm representative for NDA-related inquiries.

4. Troubleshooting Tips

This section provides guidance on troubleshooting Solidigm™ Data Center SSDs using the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool to identify issues that may impact performance or healthy monitoring.

4.1 Solidigm™ Vendor Unique SMART Log – CA Log

- "**C7 (CRC Error Count)**" Count

The "C7 (CRC Error Count)" tracks the total number of PCIe Interface CRC errors, as defined by the PCIe Link Performance Counter Parameter for "Bad TLP."

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -sensor -ssd 0 |grep Crc
```

Sample Output:

```
CrcErrorCount : 0
```

Using the NVMe-CLI tool:
Example:

```
# nvme solidigm smart-log-add /dev/nvme1 | grep "KEY\|crc"
```

Sample Output:

```
ID        KEY                  Normalized    Raw
0xc7    crc_error_count                100      0
```

Note: CRC errors may originate from the PCIe link rather than the SSD itself. If CRC errors were previously recorded, compare the count between test cases to detect increases, which may indicate link issues.

4.2 Thermal Throttling Status

- "**Thermal Throttle Status**" Count

Thermal throttling can cause performance issues in SSDs. The "Thermal Throttle Status" (EA) count indicates whether thermal throttling events have occurred.

Using Solidigm™ Storage Tool (SST):
Example:

```
# sst show -ssd 0 -smart |grep ThrottlingEventCount
```

Sample Output:

```
ThrottlingEventCount : 0
```

Using the NVMe-CLI tool:
Example:

```
# nvme solidigm smart-log-add /dev/nvme1 |grep "KEY\|thermal"
```

Sample Output:

```
ID        KEY                  Normalized    Raw
0xea    thermal_throttle_status              100      0
```

Note: If the Thermal Throttle Status (EA) count increases between test cases, investigate potential issues with the SSD or the server's fan speed control.

4.3 PCIe Link Speed and Link Width

- "**PCIe Link Speed/PCIe Link Width**" information

PCIe link stability issues can lead to performance-related problems in Solidigm™ Data Center SSDs, affecting healthy monitoring and operation. Checking the PCIe link speed and link width helps identify potential compatibility or signal integrity issues.

Using Solidigm™ Storage Tool (SST):
Run the following command to verify the PCIe link speed and width for a specific SSD:
Example:

```
# sst show -a -ssd 0 |grep Link
```

Sample Output:

```
PCILinkGenSpeed : 4
PCILinkWidth : 4
```

Using the NVMe-CLI tool:
Run the following command to retrieve PCIe link speed and width:
Example:

```
# nvme solidigm id-ctrl /dev/nvme0n1 |grep "Speed\|Wdth"
```

Sample Output:

```
linkSpeed : 4
negLnkWdth: 4
```

Using Linux Native Command:
Alternatively, use the following command to check PCIe link speed and width:

```
# lspci -s 68:00.0 -vvv |grep "LnkCap\|LnkSta"
```

Sample Output:

```
LnkCap: Port #0, Speed 32GT/s, Width x4, ASPM not supported
LnkSta: Speed 16GT/s (downgraded), Width x4 (ok)
LnkCap2: Supported Link Speeds: 2.5-32GT/s, Crosslink- Retimer+ 2Retimers+ DRS+
LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete+ EqualizationPhase1+
```

"Speed 16GT/s" means your drive is running at PCIe Gen4.0 (PCIe Gen5.0 will be 16GT/s). "Width x4" means your SSD and backplane work in PCIe X4 width. If it drops to "x2", it means the SSD's Link Width drop to "x2" and you are expecting to get half the performance.

Interpretation:

- A "Speed 16GT/s" indicates the SSD is operating at PCIe Gen 4.0.
- A "downgraded" was PCIe Gen5 SSD installed into PCIe Gen4 Slot.
- A "Width x4" confirms the SSD and backplane are using PCIe x4 width, ensuring optimal performance.
- If the width drops to "x2," performance may be reduced by half, indicating a potential issue with the SSD or server platform compatibility.

These metrics help troubleshoot performance issues, such as signal integrity or platform compatibility problems, supporting healthy monitoring.

4.4 Retrieve Telemetry Log

When SMART attributes alone cannot identify the root cause of an issue, providing telemetry logs to a Solidigm support representative can aid in troubleshooting and support healthy monitoring of Solidigm™ Data Center SSDs.

Using Solidigm™ Storage Tool (SST):
Run the following command to retrieve the telemetry log for a specific SSD:
Example:

```
# sst dump telemetry_log.bin -ssd 0 -telemetrylog
```

Sample Output:

```
- TelemetryLog BTAX135003JL7P6DGN -
Status : Successfully written TelemetryLog data to telemetry_log.bin
```

Using the NVMe-CLI tool:
Run the following command to retrieve a comprehensive telemetry log:
Example:

```
# nvme telemetry-log /dev/nvme0 –output-file=telemetry.bin
```

Sample Output:

```
# ls -al telemetry_log.bin
-rw-r--r--. 1 root root 8484352 Jun 12 16:24 telemetry_log.bin
```

Specifying Data Areas
You can specify which data area of the telemetry log to retrieve:

- Data Area 1: Contains the telemetry log header and basic controller information.
- Data Area 2: Includes additional controller state data, often more detailed than Area 1.
- Data Area 3: Typically the most comprehensive, including detailed debug information for failure analysis.
- Data Area 4: Optional, vendor-specific, and may not be supported by all devices.

Example (Data Area 3):

```
# nvme telemetry-log /dev/nvme0 --output-file=telemetry_log.bin --data-area=3
```

Specifying Output Format
You can specify the command's output format (note: the telemetry log itself remains binary):

- normal: Human-readable output for command status.
- json: Structured JSON output for scripting or automation.
- binary: Raw binary output of the command status (rarely used).

Example (JSON format):

```
# nvme telemetry-log /dev/nvme0 --output-file=telemetry_log.bin --output-format=json
```

Retrieves the Persistent Event Log
Run the following command to retrieve the persistent event log:
Example:

```
# nvme persistent-event-log /dev/nvme0 --action=read --output-file=event_log.bin
```

Note: For detailed guidance on retrieving binary logs with NVMe-CLI, refer to Retrieve Binary Logs With NVMe*-

CLI. Alternatively, use the Solidigm™ Storage Tool (SST) to extract various logs from Solidigm™ SSDs for debugging, as described in Retrieve Solidigm™ (Formerly Intel®) Solid State Drives Binary Logs With Storage Tool.

5. Manageability

This section outlines management tasks, such as firmware updates, to maintain optimal performance and support healthy monitoring of Solidigm™ Data Center SSDs using the Solidigm™ Storage Tool (SST) or the open-source NVMe-CLI tool.

5.1 Firmware Update and Activation with Reset

Updating to the latest firmware can resolve existing issues, prevent potential problems, and enhance SSD performance, contributing to effective healthy monitoring.

Using Solidigm™ Storage Tool (SST):
Run the following command to retrieve the telemetry log for a specific SSD: The latest SST version bundle the most recent firmware. Run the following command to update the firmware for a specific SSD, followed by a system reboot to activate it.

Example:

```
# sst load -ssd 0
# reboot
```

Using the NVMe-CLI tool:
Firmware activate or commit options:

-a 0: Store firmware in slot, no activation.
-a 1: Downloaded image replaces the image indicated by the Firmware Slot field.
-a 2: Activate existing firmware in slot on next reset.
-a 3: Activate existing firmware in slot immediately (if supported).
-a 4–7: Reserved/unsupported, may cause errors.

Example:

```
# nvme fw-download /dev/nvme0n1 -f 9CV10510_WFEM0260_signed.bin
# nvme fw-commit /dev/nvme0n1 -s 1 -a 1
```

To apply the new firmware, perform an NVMe reset

```
# nvme reset /dev/nvme0
```

Note:

- If an SSD does not support firmware update and activation without a reset, perform an NVMe reset or power cycle after the update.
- Firmware binaries require a signed NDA with Solidigm. Contact a Solidigm representative for NDA-related inquiries.

5.2 Firmware Activation without Reset

To support cloud datacenter requirements, Some Solidigm™ Data Center SSDs offer firmware activation without a reset, minimizing disruption to cloud applications and supporting seamless healthy monitoring.

Using the NVMe-CLI tool:
Confirming Support for Firmware Activation Without Reset
Run the following command to verify if an SSD supports firmware activation without a reset:
Example:

```
# nvme id-ctrl /dev/nvme0 -H |grep Activate
```

Sample Output:

```
[4:4] : 0x1   Firmware Activate Without Reset Supported
```

Note: A value of "0x1" indicates that "Firmware Activate Without Reset" is enabled.

Checking Number of Firmware Slots
NVMe devices typically support 1–7 firmware slots, each holding a firmware image, with one slot active at a time. Slot 0 is often reserved for the controller to automatically select a slot.
Run the following command to check the number of supported firmware slots:
Example:

```
# nvme id-ctrl /dev/nvme0 -H |grep "Number of Firmware Slots"
```

Sample Output:

```
[3:1] : 0x4   Number of Firmware Slots
```

Note: A value of "0x4" indicates the SSD support 4 firmware slots.

Updating and Activating Firmware Without Reset
To update and activate new firmware without a reset, use the following commands:
Example:

```
# nvme fw-download /dev/nvme0n1 -f 9CV10510_WFEM0260_signed.bin
# nvme fw-commit /dev/nvme0n1 -s 1 -a 3
```

5.3 Change and Verify Power Modes

Solidigm™ Data Center SSDs support a power governor feature, allowing users to adjust power modes to optimize performance and power consumption, which supports healthy monitoring. These SSDs typically offer 3–5 power modes, detailed in the product specification.

Note: Access to the product specification requires a signed NDA with Solidigm. Contact a Solidigm representative for NDA-related inquiries. Alternatively, use the NVMe-CLI command to view power mode details without an NDA:
Example:

```
# nvme id-ctrl /dev/nvme1 |grep -w  ps
```

Sample Output:

```
ps     0 : mp:25.00W operational enlat:30000 exlat:30000 rrt:0 rrl:0
ps     1 : mp:20.00W operational enlat:30000 exlat:30000 rrt:1 rrl:1
ps     2 : mp:17.00W operational enlat:30000 exlat:30000 rrt:2 rrl:2
ps     3 : mp:14.00W operational enlat:30000 exlat:30000 rrt:3 rrl:3
ps     4 : mp:5.00W non-operational enlat:30000 exlat:30000 rrt:4 rrl:4
```

Using the NVMe-CLI tool:
Change Power Mode
Run the following command to view the current power mode for a specific SSD:
Example:

```
# sst show -d PowerGovernorMode -ssd 0
```

Sample Output:

```
- BTAX135003JL7P6DGN 1 -
PowerGovernorMode : 0
```

Setting Power Mode

Run the following command to set the power mode, followed by a system reboot to apply the change:
Example:

```
# sst set -ssd 0 PowerGovernorMode=0
# reboot
```

Sample Output:

```
Set PowerGovernorMode successful. Completed successfully.
```

Using the NVMe-CLI tool:
Change Power Mode
Run the following command to check the current power management setting:
Example:

```
# nvme get-feature /dev/nvme1n1 -f 0x2h
```

Sample Output:

```
get-feature:0x02 (Power Management), Current value:00000000
```

Setting Power Mode
Run the following command to set the power mode, followed by an NVMe reset to apply the change:
Example:

```
# nvme set-feature /dev/nvme1 -f 0x2h -v 0
# nvme reset /dev/nvme1
```

Sample Output:

```
set-feature:0x02 (Power Management), value:00000000, cdw12:00000000, save:0
```

Using NVMe-CLI with OCP Plugin
Per the OCP Specification, Data Center SSD (DSSD) power states can be managed using the Set/Get Features function with Feature Identifier C7h, accessible via the NVMe-CLI OCP plugin.

Checking DSSD Power State
Run the following command to view the current power state:
Example:

```
# nvme ocp get-dssd-power-state-feature /dev/nvme2 –all
```

Sample Output:

```
get-feature:0xC7 Current value: 0x000019
get-feature:0xC7 Default value: 0x000019
get-feature:0xC7 Saved value: 0x000019
```

Note: The hexadecimal value 0x000019 corresponds to a decimal value of 25, indicating 25W.

Setting and Verifying DSSD Power State
Run the following command to set the power state:
Example:

```
# nvme ocp set-dssd-power-state-feature /dev/nvme1 --power-state=0x000014
```

Sample Output:

```
Successfully set DSSD Power State (feature: 0xC7) to below values
DSSD Power State: 0x14
Save bit Value: 0x0
```

To verify the change, run the following command:
Example:

```
# nvme ocp get-dssd-power-state-feature /dev/nvme1 –all
```

Sample Output:

```
get-feature:0xC7 Current value: 0x000014
get-feature:0xC7 Default value: 0x000019
get-feature:0xC7 Saved value: 0x000019
```

Note: The hexadecimal value 0x000014 corresponds to a decimal value of 20, indicating 20W.

5.4 Format the drive to different VSS

Solidigm™ Data Center SSDs support variable sector sizes (VSS) through Extended LBA Format, allowing customization of sectors sizes to meet application needs and ensuring optimal data handling.

Confirming Supported Variable Sector Sizes
Use the NVMe-CLI tool to check the supported sector sizes for an SSD:

Example:

```
# nvme id-ns /dev/nvme0n1 |grep -w lbaf
```

Sample Output:

```
lbaf  0 : ms:0   lbads:9  rp:0x2 (in use)
lbaf  1 : ms:8   lbads:9  rp:0x2
lbaf  2 : ms:0   lbads:12 rp:0x2
lbaf  3 : ms:8   lbads:12 rp:0x2
lbaf  4 : ms:64  lbads:12 rp:0x2
```

Interpretation:
The SSD supports the following 512, 520, 4096, 4104, 4160 bytes host sector sizes using the following LBA formats:

- LBA Format 0 (512B): LBA Data Size = 512B, Metadata Size = 0
- LBA Format 1 (520B): LBA Data Size = 512B, Metadata Size = 8B
- LBA Format 2 (4096B): LBA Data Size = 4096B, Metadata Size = 0
- LBA Format 3 (4104B): LBA Data Size = 4096B, Metadata Size = 8B
- LBA Format 4 (4160B): LBA Data Size = 4096B, Metadata Size = 64B

Note: Detailed VSS information is available in the SSD product specification, which requires a signed NDA with Solidigm. Contact a Solidigm representative for NDA-related inquiries.

Formatting the Drive Using Solidigm™ Storage Tool (SST)
Run the following command to format an SSD to a specific LBA format (e.g., 520B with LBA Format 1):

Example:

```
# sst start -ssd 0 -nvmeformat lbaformat=1 secureerasesetting=0
```

Sample Output:

```
WARNING! You have selected to format the drive!
Proceed with the format? (Y|N): y
Formatting...(This can take several minutes to complete)

- Intel SSD DC P5520 Series BTAX135003JL7P6DGN -

Status : NVMeFormat successful.
```

Formatting the Drive Using NVMe-CLI Tool
Run the following command to format an SSD to a specific LBA format (e.g., 4104B with LBA Format 3):

Example:

```
# nvme format /dev/nvme0n1 --lbaf=3 --ses=0
```

Sample Output:

```
You are about to format nvme0n1, namespace 0x1.
WARNING: Format may irrevocably delete this device's data.
You have 10 seconds to press Ctrl-C to cancel this operation.

Use the force [--force] option to suppress this warning.
Sending format operation ...
Success formatting namespace:1
```

To verify the format change:

Example:

```
# nvme list |grep nvme0
```

Sample Output:

```
/dev/nvme0n1   … SSDPF2KX076T1   0x1   7.68  TB /   7.68  TB    4 KiB +  8 B   9CV10510
```

Note: The NVMe-CLI tool does not support SATA drives. For SATA drives, use the SST tool to perform similar formatting functions.

5.5 Sanitize the SSD

Solidigm™ Data Center SSDs support NVMe sanitize operations, which securely erase all user data, including residual data in over-provisioned areas, restoring the drive to a factory-like state. This process supports healthy monitoring by ensuring data security and drive readiness for redeployment.

NVMe Sanitize Types:

- Block Erase: Physically erases all flash blocks.
- Overwrite: Writes patterns (e.g., zeros, ones, random data) to all areas.
- Crypto Erase: Resets the encryption key if the drive supports self-encryption, rendering data unreadable.

Checking Supported Sanitize Options

Use the NVMe-CLI tool to verify which sanitize operations are supported by the SSD:

Example:

```
# nvme id-ctrl /dev/nvme0 -H |grep Sanitize
```

Sample Output:

```
[29:29] : 0x1 No-Deallocate After Sanitize bit in Sanitize command Not Supported
[2:2] : 0     Overwrite Sanitize Operation Not Supported
[1:1] : 0x1   Block Erase Sanitize Operation Supported
[0:0] : 0x1   Crypto Erase Sanitize Operation Supported
```

Note: A value of "0x1" indicates support for the respective sanitize operation (e.g., Block Erase and Crypto Erase in this example).

Run the Sanitize Commands

- Block Erase

To perform a Block Erase sanitize operation and verify its status via sanitize log

Examples:

```
# nvme sanitize /dev/nvme0n1 --sanact=2
# nvme sanitize-log /dev/nvme0n1
```

Sample Output:

```
Sanitize Progress                   (SPROG) :  65535
```

```
Sanitize Status                 (SSTAT) :  0x101
Sanitize Command Dword 10 Information (SCDW10) :  0x2
Estimated Time For Overwrite              :  4294967295 (No time period reported)
Estimated Time For Block Erase            :  90
Estimated Time For Crypto Erase           :  13
Estimated Time For Overwrite (No-Deallocate)   :  0
Estimated Time For Block Erase (No-Deallocate) :  0
Estimated Time For Crypto Erase (No-Deallocate):  0
Estimated Time For Post-Verification Deallocation:  0
Sanitize State Information          (SSI) : 0
```

- Crypto Erase

To perform a Crypto Erase sanitize operation and verify its status via sanitize log

Examples:

```
# nvme sanitize /dev/nvme0n1 --sanact=4
# nvme sanitize-log /dev/nvme0n1
```

Sample Output:

```
Sanitize Progress               (SPROG) :  65535
Sanitize Status                 (SSTAT) :  0x101
Sanitize Command Dword 10 Information (SCDW10) :  0x2
Estimated Time For Overwrite              :  4294967295 (No time period reported)
Estimated Time For Block Erase            :  90
Estimated Time For Crypto Erase           :  13
Estimated Time For Overwrite (No-Deallocate)   :  0
Estimated Time For Block Erase (No-Deallocate) :  0
Estimated Time For Crypto Erase (No-Deallocate):  0
Estimated Time For Post-Verification Deallocation:  0
Sanitize State Information          (SSI) : 0
```

5.6 Device Self-Test

The NVMe Device Self-Test (DST) is a diagnostic feature defined in the NVMe specification that allows an NVMe drive to perform internal tests to verify the integrity and functionality of the controller and, in some cases, the storage media. Solidigm™ Datacenter SSD supports DST with two main types: a short test (typically under 2 minutes) and an extended test (longer duration, may involve reading/writing to media).

Importantly, the DST tests are designed to preserve user data and allow the drive to remain operational, either by running in the background or pausing to service I/O requests.

Use the NVMe-CLI tool to confirm if an SSD supports DST:

Examples:

```
# nvme id-ctrl /dev/nvme0n1 -H |grep Self-test
```

Sample Output:

```
[4:4] : 0x1   Device Self-test Supported
```

Note: A value of "0x1" indicates that DST is supported.

To perform a short DST and check its progress:

Examples:

```
# nvme device-self-test /dev/nvme0 -s 1
# nvme self-test-log /dev/nvme0 |grep Completion
```

Sample Output:

```
Short Device self-test started
Current Completion : 0%
```

To perform an extended DST and check its progress:

Examples:

```
# nvme device-self-test /dev/nvme0 -s 2
# nvme self-test-log /dev/nvme0 |grep Completion
```

Sample Output:

```
Short Device self-test started
Current Completion : 20%
```

5.7 Latency Monitor

Per the OCP Specification, Latency Monitor (Log Identifier C3h) was introduced for DSSD in order to troubleshoot performance related issues. This feature will allow suppliers and hyperscale companies to clearly understand if the outliers are caused by the SSD or other components in the Host system.

Using Solidigm™ Storage Tool (SST)

Run the following command to retrieve latency statistics for read commands on a specific SSD:

Example:

```
# sst show -ssd 0 -latencystatistics reads
```

Sample Output:

```
- Latency Statistics For Read Commands BTAX135003JL7P6DGN -

Major Version : 4
Minor Version : 9
Latency Bucket Details : (Standard Latency Stats)

- Buckets -

Bucket 1, 0.0000 - 1.0000 us : 0
Bucket 2, 1.0000 - 2.0000 us : 0
Bucket 3, 2.0000 - 3.0000 us : 0
Bucket 4, 3.0000 - 4.0000 us : 0
Bucket 5, 4.0000 - 5.0000 us : 0
Bucket 6, 5.0000 - 6.0000 us : 0
Bucket 7, 6.0000 - 7.0000 us : 0
Bucket 8, 7.0000 - 8.0000 us : 0
Bucket 9, 8.0000 - 9.0000 us : 0
…
```

Using NVMe-CLI Tool with OCP Plugin

Run the following command to retrieve the Latency Monitor log (C3h) for an SSD:

Example:

```
# nvme ocp latency-monitor-log /dev/nvme1n1
```

Sample Output:

```
NVMe Status:Successful Completion: The command completed without error(0)
-Latency Monitor/C3 Log Page Data-
  Controller   :  nvme1n1
  Feature Status                0x7
  Active Bucket Timer           2230 min
  Active Bucket Timer Threshold     10080 min
  Active Threshold A            30 ms
```

Active Threshold B	100 ms
Active Threshold C	155 ms
Active Threshold D	235 ms
Active Latency Configuration	0xfff
Active Latency Minimum Window	1000 ms
Active Latency Stamp Units	4095
Static Latency Stamp Units	4095
Debug Log Trigger Enable	4032
Debug Log Measured Latency	0
Debug Log Latency Time Stamp	1970-01-01D|00:00:00:000
Debug Log Pointer	0
Debug Counter Trigger Source	0
Debug Log Stamp Units	1
Log Page Version	1
Log Page GUID	85D45E58D4E643709C6C84D08CC07A92

	Read	Write	Deallocate/Trim
Active Bucket Counter: Bucket 0	0	0	0
Active Bucket Counter: Bucket 1	0	0	0
Active Bucket Counter: Bucket 2	0	0	0
Active Bucket Counter: Bucket 3	1	0	0
Active Latency Time Stamp: Bucket 0	N/A	N/A	N/A
Active Latency Time Stamp: Bucket 1	N/A	N/A	N/A
Active Latency Time Stamp: Bucket 2	N/A	N/A	N/A
Active Latency Time Stamp: Bucket 3	N/A	N/A	N/A
Active Measured Latency: Bucket 0	0 ms	0 ms	0 ms
Active Measured Latency: Bucket 1	0 ms	0 ms	0 ms
Active Measured Latency: Bucket 2	0 ms	0 ms	0 ms
Active Measured Latency: Bucket 3	2288 ms	0 ms	0 ms

Static Bucket Counter: Bucket 0	0	0	0
Static Bucket Counter: Bucket 1	0	0	0
Static Bucket Counter: Bucket 2	0	0	0
Static Bucket Counter: Bucket 3	0	0	0
Static Latency Time Stamp: Bucket 0	N/A	N/A	N/A
Static Latency Time Stamp: Bucket 1	N/A	N/A	N/A
Static Latency Time Stamp: Bucket 2	N/A	N/A	N/A
Static Latency Time Stamp: Bucket 3	N/A	N/A	N/A
Static Measured Latency: Bucket 0	0 ms	0 ms	0 ms
Static Measured Latency: Bucket 1	0 ms	0 ms	0 ms
Static Measured Latency: Bucket 2	0 ms	0 ms	0 ms
Static Measured Latency: Bucket 3	0 ms	0 ms	0 ms

6. Capability Checking

This section describes methods to verify feature support and device capabilities for Solidigm™ Data Center SSDs using the NVMe-CLI tool, aiding in troubleshooting and supporting healthy monitoring by ensuring proper configuration and functionality.

6.1 Device OCP Capabilities Information

Per the OCP Specification, the Device Capabilities log (Log Identifier C4h) provides a consolidated report of critical device-specific support information for Solidigm™ Data Center SSDs (DSSD), supporting healthy monitoring by detailing feature and configuration support.

Retrieving Device Capabilities Log
Use the NVMe-CLI tool with the OCP plugin to retrieve the Device Capabilities log:
Example:

```
# nvme ocp device-capability-log /dev/nvme1
```

Sample Output:

```
Device Capability/C4 Log Page Data
PCI Express Ports                          : 0x1
OOB Management Support                     : 0x8007
Write Zeroes Command Support               : 0x801f
Sanitize Command Support                   : 0x8017
Dataset Management Command Support         : 0x8003
Write Uncorrectable Command Support        : 0x8007
Fused Operation Support                    : 0x8001
Minimum Valid DSSD Power State             : 0x5
DSSD Power State Descriptors               :
0x0000848484848484848484838383828282818181818180808080808080808080808080808080808080808080808080808080808080
808080808080808080808080808080808080808080808080808080808080808080808080808080808080808080808080808080808
0808080808080808080808080808080808080808080
Log Page Version                           : 0x1
Log page GUID                              : 0xb7053c914b58495d98c9e1d10d054297
```

Note: Refer to the OCP Specification for detailed value interpretations. For example, a "Minimum Valid DSSD Power State" value of 0x5 indicates that the SSD (/dev/nvme1) supports five power states.